Algorytmy równoległe

Sekwencyjne mnożenie macierzy

Marcin Rociek
Informatyka, IV rok

25 października 2001

1 Założenia

Badanie sekwencyjnego mnożenia macierzy przeprowadzone zostało dla różnych typów danych przechowywanych w macierzach, różnych sposobach przechowywania macierzy, przeglądania macierzy oraz wreszcie dla różnych kompilatorów i systemów operacyjnych.

1.1 Typy danych

Danymi przechowywanymi w macierzach były liczby typu `float` (4 bajty) oraz `double` (8 bajtów). Ponieważ czasy mnożenia macierzy składających się z danych typu `int` (4 bajty) były zbliżone do czasów mnożenia macierzy liczb `float`, ograniczono się więc tylko do tych dwóch wyżej wymienionych typów.

1.2 Przechowywanie macierzy w pamięci

Generalnie macierze były przechowywane wedle takiej zasady, w której znajdujące się obok siebie liczby z jednego wiersza macierzy znajduja się w pamięci obok siebie. Dla dynamicznego przydzielania pamięci rozpatrzone zostały dwa przypadki: 1) jedna duża tablica o rozmiarach $n \times n$; 2) n-elementowa tablica wskaźników do n-elementowych tablic zawierających liczby z jednego wiersza. W pierwszym przypadku miejsce w tablicy, w którym znajduje się dana liczba (indeks) określamy sami jako $y \cdot n + x$. Przy mnożenia macierzy, gdzie odczyt/zapis następuje do kolejnych miejsc w wierszu/kolumnie można zamiast mnożenia w celu wyznaczenia indeksu stosować po prostu jego odpowiednie zwiększanie. W drugim przypadku do danej liczby dostajemy się po prostu poprzez `m[y][x]`.

Podobne dwa przypadki zbadane zostały dla statycznego przydzielania pamięci, z tym że wielkości mnożonych macierzy ustalane były już na etapie kompilacji:

```
const int size = 500;
float m[size*size];           // w przypadku tablicy jednowymiarowej
float m[size][size];          // w przypadku tablicy dwuwymiarowej
```

Ponadto przyjęto założenie, że wszystkie trzy macierze (tj. wejściowe i wyjściowa) są przechowywane w jednakowy sposób.

1.3 Przeglądanie macierzy

Macierz wynikowa tworzona była dwoma różnymi sposobami: poziomo (wtedy zapisywane są kolejne komórki pamięci, znajdujące się w jednym wierszu) lub pionowo. Stosowanie dwóch metod przeglądania macierzy eliminuje konieczność sprawdzania innych niż pozioma metod przechowywania macierzy.

1.4 Systemy operacyjne i kompilatory

Testy przeprowadzono w dwóch różnych środowiskach: w systemie Windows, gdzie użytym kompilatorem był **cl** (z Microsoft Visual C++) oraz w systemie Linux, gdzie kod kompilowany był przy pomocy **g++**.

Należy przy tym zaznaczyć, że w przypadku obu kompilatorów istotne okazało się włączanie opcji optymalizujących (`-O3` dla **gcc** i `-Ox` dla **cl**), gdyż ich brak powodował *kilkukrotne* wydłużenie czasu działania programu.

2 Wyniki dla macierzy 500×500 i 1000×1000

W tabeli 1 zestawiono czasy obliczeń (w sekundach), dla macierzy 500×500 i 1000×1000 dla każdego przypadku typu danych itp. Pogrubioną czcionką zaznaczone są wyniki najlepsze.

Zaskakujące są niekorzystne wyniki otrzymane pod Linuxem dla małej macierzy liczb `float`, oraz wyniki otrzymane pod Windowsami dla dużej macierzy liczb `double`. W tym drugim przypadku jeszcze dziwniejsze jest dwukrotne wydłużenie czasu obliczeń przy wykorzystaniu dynamicznie zaalokowanego bloku pamięci!

				float		**double**	
(1)	**(2)**	**(3)**	**(4)**	**500×500**	**1000×1000**	**500×500**	**1000×1000**
W	D	A	H	3,15	34,14	5,7	127,74
W	D	A	V	4,15	35,55	8,2	108,88
W	D	V	H	5,37	48,8	8,07	80,73
W	D	V	V	4,3	38,38	9,23	82,99
W	S	A	H	3,15	36,82	5,4	65,4
W	S	A	V	4,18	38,13	4,99	61,82
W	S	V	H	**3,14**	37,46	5,49	68,93
W	S	V	V	4,20	35,29	**4,84**	59,76
L	D	A	H	l 3,7	**30,82**	4,97	**49,48**
L	D	A	V	4,14	35,03	6,46	51,34
L	D	V	H	6,11	51,77	9,57	77,66
L	D	V	V	5,03	42,14	6,94	58,89
L	S	A	H	4,67	39,44	6,31	81,26
L	S	A	V	5,06	43,68	7,79	87,52
L	S	V	H	4,55	38,98	6,31	76,93
L	S	V	V	5,27	44,65	7,78	88,27

Tablica 1: Porównanie czasu obliczeń dla różnych sposobów reprezentacji danych, kolejności ich przeglądania itp. Kolumna (1) to system operacyjny (i co za tym idzie - kompilator): W - Windows (cl), L - Linux (gcc); kolumna (2) to sposób alokacji pamięci: D - dynamicznie, S - statycznie; kolumna (3) to sposób reprezentacji macierzy: A - jedna duża tablica $n \times n$, V - tablica n wektorów o rozmiarze n; kolumna (4) to kolejność przeglądania danych: H - poziomo, V - pionowo.

3 Wykresy $t = f(n)$

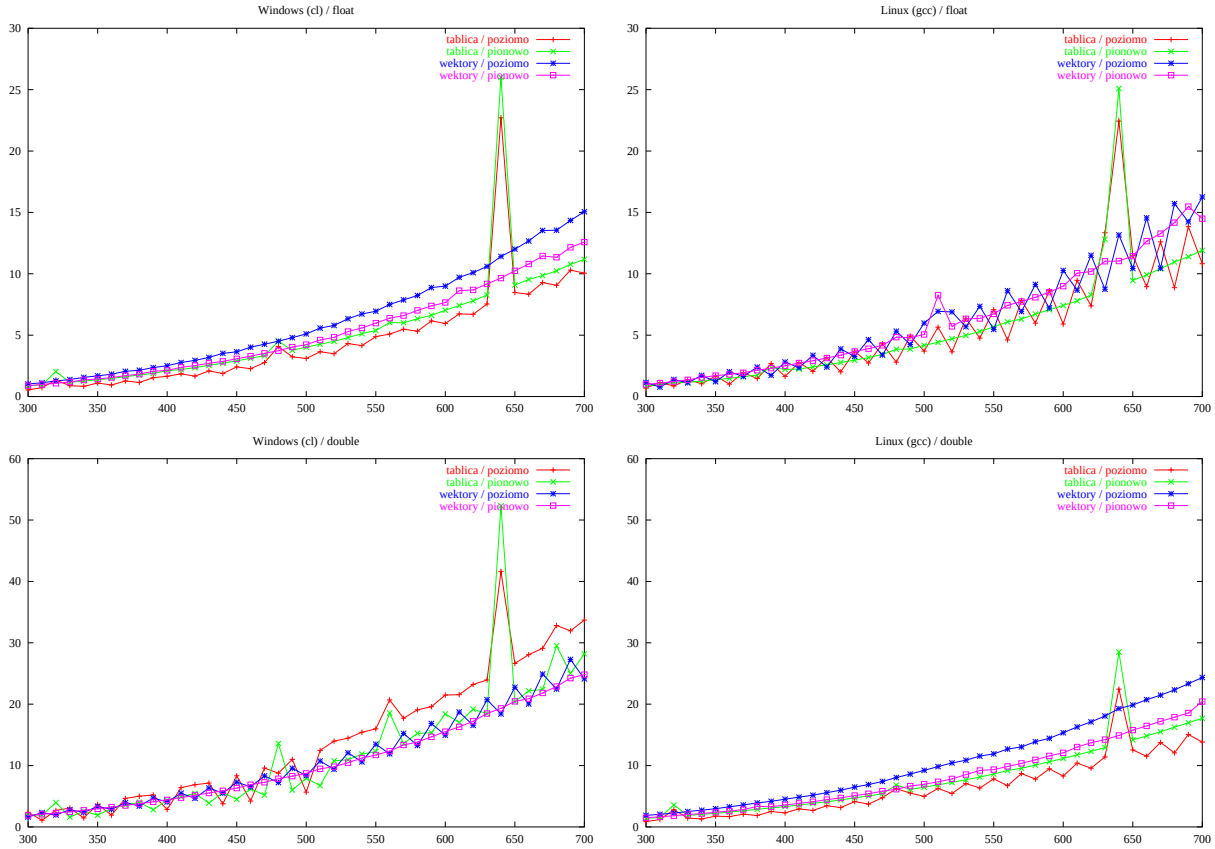
Ponieważ serie liczb podane w tablicy 1 niewiele niestety mówią, wykonano jeszcze kilka wykresów obrazujących czas obliczeń w zależności od wielkości macierzy (testy te wykonano tylko dla przypadku pamięci alokowanej dynamicznie).

Rysunek 1 przedstawia takie właśnie wykresy wykonane dla macierzy o rozmiarach od 300×300 do 700×700 z krokiem co 10. Widać, że niektóre z wykresów rosną w miarę jednostajnie, inne zaś w sposób ząbkowy. Charakterystycznym elementem jest dziwne spowolnienie obliczeń dla macierzy 640×640 występujące w każdym przypadku, gdy macierz przechowywana jest w tablicy jednowymiarowej. Poza tym szczególnym przypadkiem wydaje się, że przechowywanie macierzy w tablicy jednowymiarowej jest najlepszym rozwiązaniem (z wyjątkiem macierzy liczb `double` mnożonych pod Windowsami).

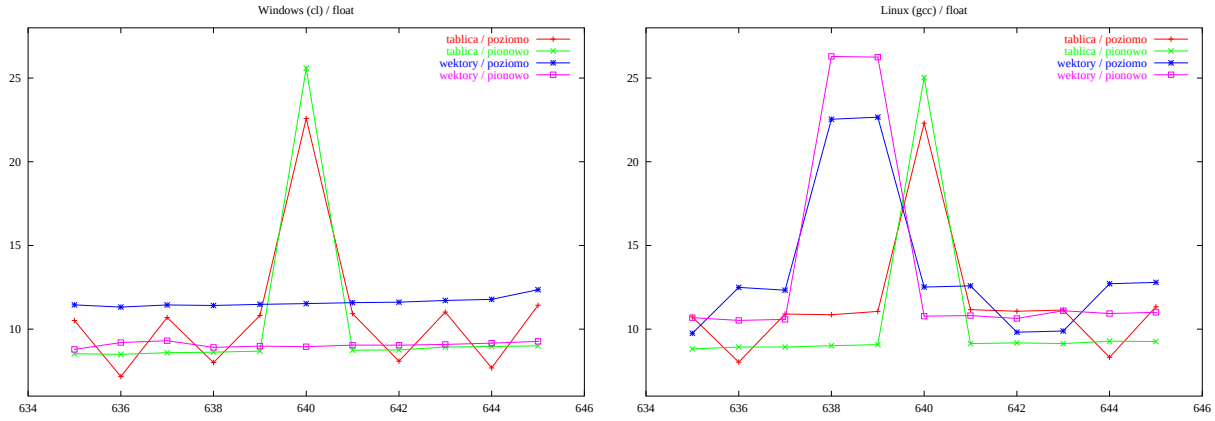
Aby dokładniej przyjrzeć się zagadkowej liczbie 640 wykonane zostały jeszcze dwa wykresy dla macierzy o rozmiarach od 635×635 do 645×645, przy czym dla zwiększenia pewności każdy wynik został wyznaczony jako średnia z 5 pomiarów. Jak widać na rysunku 2 oprócz wspomnianego spowolnienia macierzy 640×640 pojawiły się nowe spowolnienia, tym razem dla macierzy przechowywanej jako wektory, ale tylko w przypadku Linuxa.

4 Wykorzystany sprzęt i oprogramowanie

Testy przeprowadzone zostały na maszynie wyposażonej w procesor Intel Celeron 433MHz, 128KB pamięci podręcznej, 384MB pamięci RAM. Systemem operacyjnym był Windows 98 SE oraz Linux (Slackware 7, jądro 2.2.13). Do kompilacji używany był kompilator **cl 11.00.7022** (z MSVC++ 5.0) oraz **g++ egcs-2.91.66**.



Rysunek 1: Czas obliczeń (w sekundach) w zależności od wielkości macierzy (z krokiem co 10). Wyraźnie widoczne zagadkowe spowolnienia dla macierzy 640×640 przechowywanej w jednej dużej tablicy.



Rysunek 2: Czas obliczeń (w sekundach) w zależności od wielkości macierzy (z krokiem co 1). Każdy wynik jest średnią z 5 pomiarów. Ukazały się kolejne zagadkowe spowolnienia dla macierzy 638×638 i 639×639 przechowywanych w wektorach, ale tylko w przypadku Linuxa (gcc)!.